二进制漏洞挖掘系列(8)-初识格式化字符串漏洞

WritenBy 東

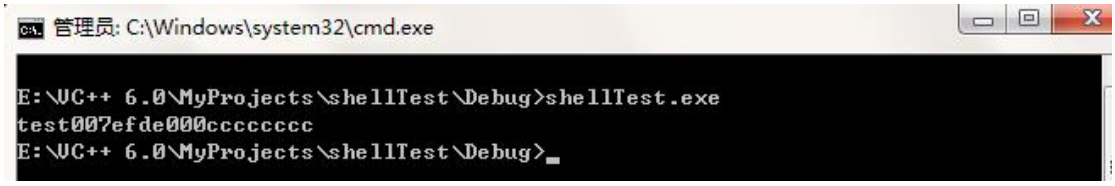
提到格式化字符串漏洞,有的同学可能就比较陌生了,但是接触过编程的同学知道 C 语言有格式化输出 printf(“%d”,num); 所以提到格式化字符串漏洞肯定是和带有这类格式输出的函数有关系,类似这样的函数 **printf ,sprintf,**fprintf,vprintf..等等 C 语言库函数中的 printf 家族

那什么是格式化字符串漏洞呢?我们先来看几个例子

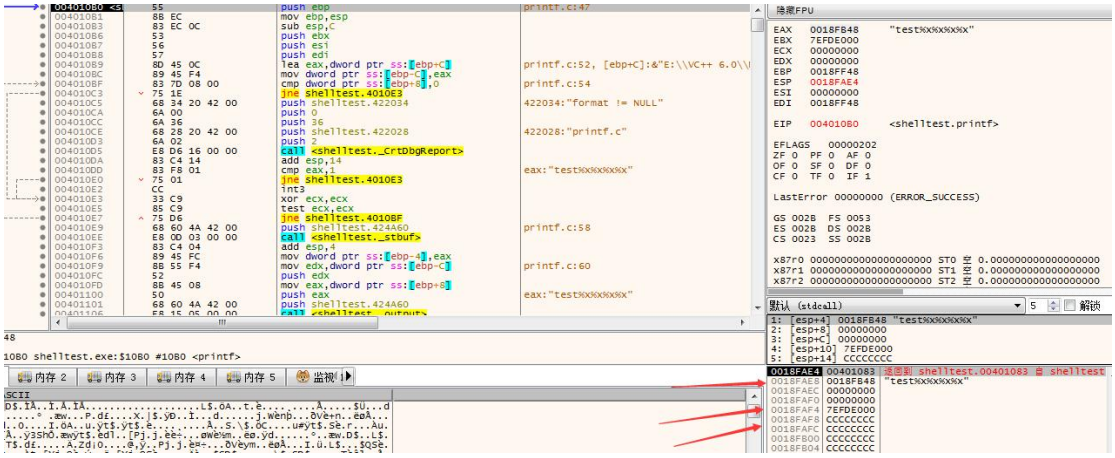
Eg.1

```
#include <stdio.h>
#include "windows.h"
int main(int argc,char **argv)
{
	char buf[1024];
	char ove[]="test%x%x%x%x";
	strncpy(buf,ove,sizeof(buf)-1);
	printf(buf);
	return 0;
}
```

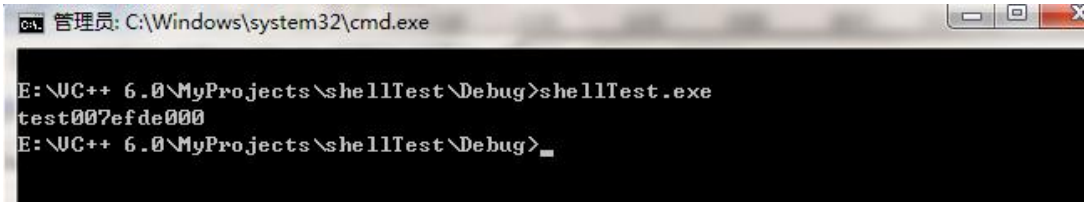
我们发现会输出一些额外的数据



接着用调试器跟踪查看一下 printf 函数的栈帧 发现打印出其他的数据就是栈底数据



当输入参数包含格式符号时导致打印出栈上的数据出来,进一步确认我们把 ove[]数组改一下改为 char ove[]=”test%x%x%x”发现他并没有把 CCCCCCCC 打印出来说明了确实是打印出了栈数据

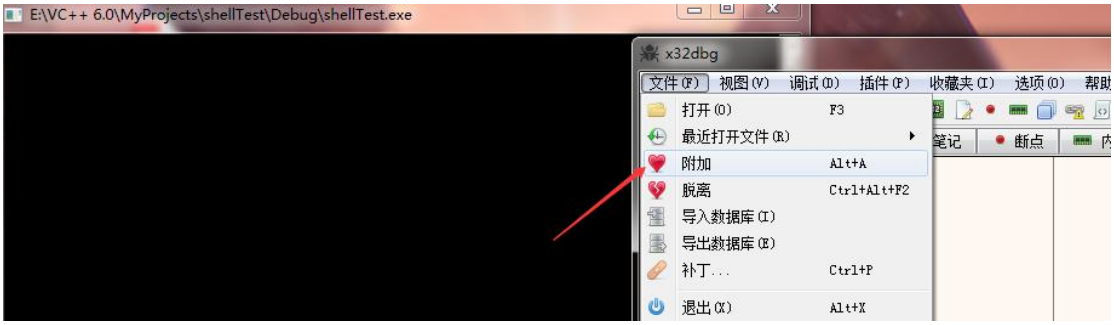


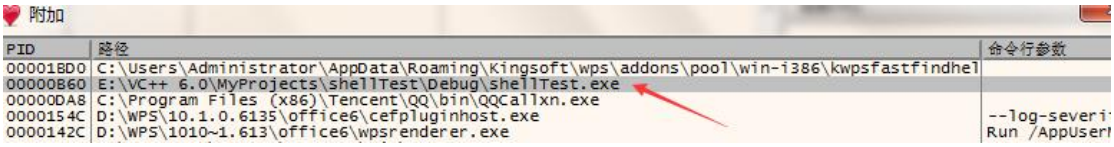
回到上面观察函数栈帧,我们给 printf 函数传了一个参数进去 程序会默认的将栈中的下一个数据(我这个是下两个数据..)作为参数传给 printf 函数 由于栈中下个数据正好是 strncpy 的目标地址 buf 它刚好指向 test%x%x%x 地址 7efde000 所以程序就输出了 007efde000 我这个可能有个函数的 check cookies 校验 如果想接着打印出他的参数就接着加%x 从而实现遍历栈上的数据

格式化字符串的利用:

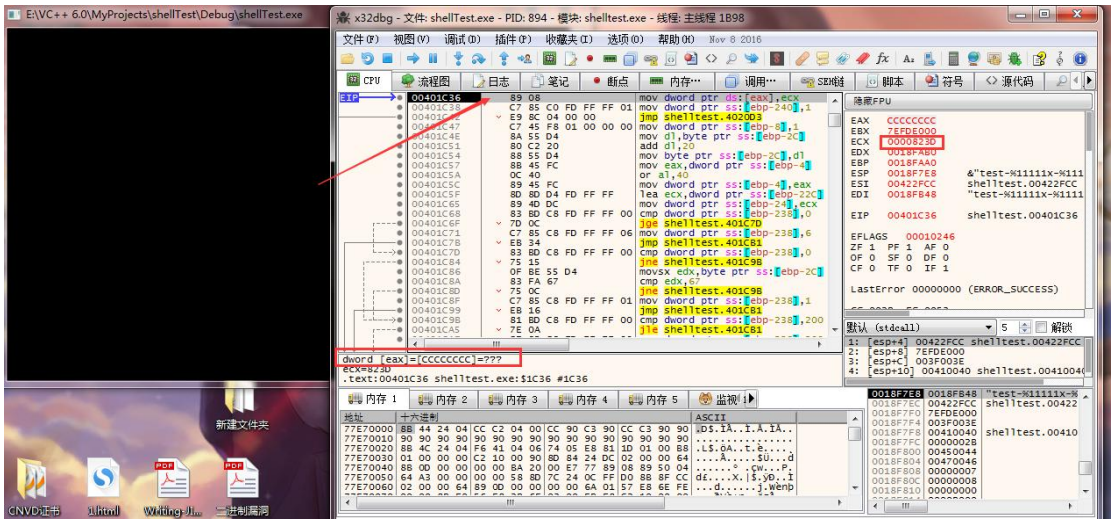
接着上面我们在后面多输入%n 意思为回车或换行,在反汇编中实际是一个写的操作
由于%x 可以控制输出字符个数接着我们稍微更改一下上面的程序
char ove[]="test-%11111x-%11111x-%11111x-%n";
再加一个 getchar 方便用于调试器附加

```
int main(int argc,char **argv)
{   char buf[1024];
    char ove[]="test-%11111x-%11111x-%11111x-%n";
    getchar();
    strncpy(buf,ove,sizeof(buf)-1);
    printf(buf);
    return 0;
}
```

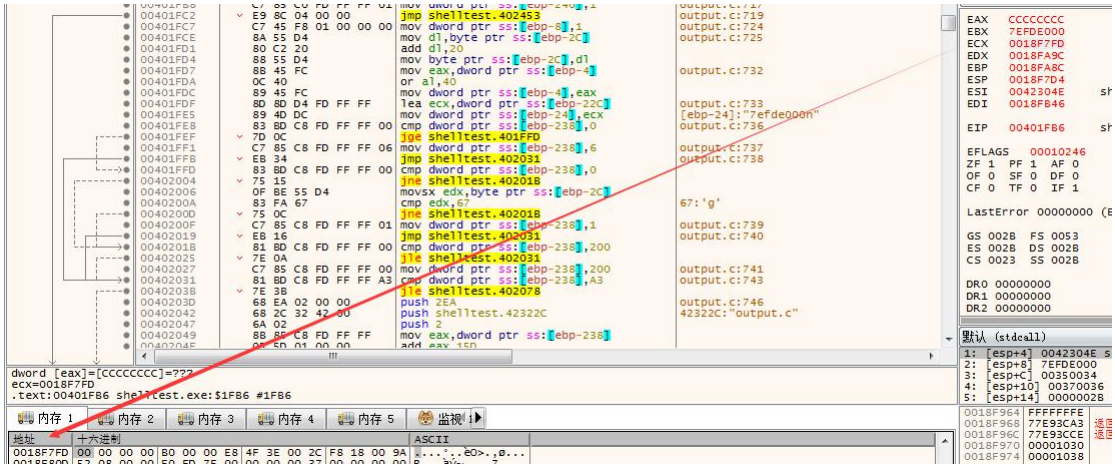




打开程序,在等待输入这里附加进程,然后输入个字符 回车,按键触发漏洞了发现程序断在了这个 mov eax ecx 这里, 该指令将 ecx 写入 eax 是就是%n 格式化字符在反汇编中导致了写操作



0018F7E8 即十进制的 1636328 由于 ecx 是可控的,我们把他指向 0018F7E8 附近的地址,前面是三个格式控制输出所以 1636328/3=545442 为了方便观察我们在 test 后加一串 AAAA...重启附加,观察这次的 ecx 指向 位置貌似是变了,

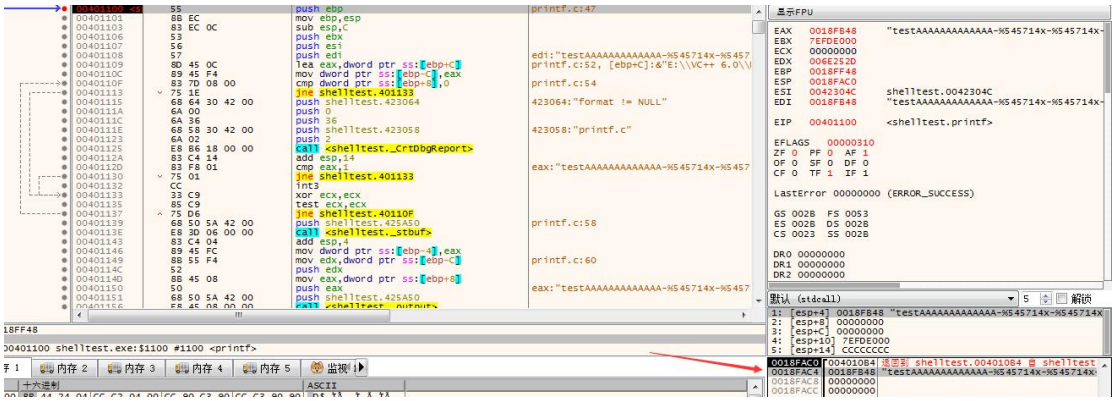


像下翻看到了我们的字符串,重新计算一下 test 字串的地址 0018FB0D+7=0018FB14

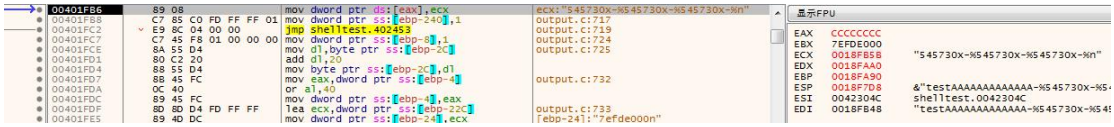
由于我们想让 ecx 指向我们的 AAA..那么 AAA..的起始地址就是 0018FB0D+B=18FB18 的十进制为 1637144, /3 =545714 修改程序为

char ove[]="testAAAAAAAAAAAAAAA-%545714x-%545714x-%545714x-%n";

重新载入



Test 字符串的地址是 0018FB48 十进制 1637192/3=545730



成功的就将数据写入了 ecx 由于 eax 作为返回值所以我们也可以控制 eax, 将返回地址改写成一个跳板,调试过程中地址总是在变,在实验中多试验几次就可以达到代码执行的效果,相对典型的缓冲区溢出漏洞 windows 上的格式化字符串漏洞利用起来还是比较难的